Zend_Service_PayPal - A.J. Brown

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Service_PayPal Component Proposal

Proposed Component Name	Zend_Service_PayPal
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Service_PayPal
Proposers	A.J. Brown, Zend liaison
Revision	0.1 - 27 February 2007: Started proposal process 0.2 - 23 March 2007: First proposal draft ready for review 0.3 - 9 June 2007: Added the MassPay API 0.4 - 17 June 2007: Added the GetTransactionDetails API 0.5 - 1 July 2007: Added PDT and IPN requirements 0.6 - 2 February 2008: Transfered proposal ownership from Shahar Evron (wiki revision: 29)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Zend_Service_PayPal is a Zend Framework interface to the PayPal on-line payment service, through PayPal's exposed APIs. Initially, it is intended to support the lighter PayPal Name-Value Pairs (NVP) interface. Additionally, Zend_Service_PayPal will provide smaller "utility" classes for sending special requests to PayPal (eg. Payment Data Transfer requests) or receiving Instant Payment Notification messages from PayPal. In the future, Zend_Service_PayPal might also implement a SOAP interface as an optional alternative to the NVP interface.

2. References

- PayPal API refenrece
- PayPal NVP API reference (PDF)
- PayPal NVP API PHP examples
- PayPal developer testing guide
- Zend_Service_Payment proposal

3. Component Requirements, Constraints, and Acceptance Criteria

- Zend_Service_PayPal **will** implement the new PayPal NVP interface to the PayPal API servers
- Zend_Service_PayPal **will** include a set of container classes for common PayPal data types: CreditCard, Address, Authentication Information, ...
- Zend_Service_PayPal **will** allow API users to authenticate using both PayPal API Signatures and PayPal API Certificates
- Zend_Service_PayPal **will** allow API users to perform the business functions:
    - DoDirectPayment
    - SetExpressCheckout
    - GetExpressCheckoutDetails
    - DoExpressCheckoutPayment
    - DoCapture
    - DoAuthorize
    - DoVoid
    - DoReauthorize
    - RefundTransaction
    - TransactionSearch
    - GetTransactionDetails
    - MassPay
- Zend_Service_PayPal **will** expose methods in a way allowing abstraction between the NVP interface and the SOAP interface if it will be implemented
- Zend_Service_PayPal **will** implement a common PayPal API response class allowing easy access to response messages
- Zend_Service_PayPal **will** implement classes to send PayPal PDT (Payment Data Transfer) requests and receive PayPal IPN (Instant Payment Notification) requests. Both functions are not a part of PayPal's NVP API, and should be implemented as separate interfaces.
- Zend_Service_PayPal **will** use Zend_Http_Client to preform HTTP requests, and will allow the developer to override the default Http_Client object (allowing the usage of custom Http_Client adapters, etc.)
- Zend_Service_PayPal **will not** attempt to be built in a payment-service abstract fashion, due to the complexity of PayPal's service

4. Dependencies on Other Framework Components

- Zend_Http_Client
- Zend_Validate_Ccnum
- Zend_Validate_Alnum
- Zend_Validate_EmailAddress
- Zend_Validate_StringLength
- Zend_Service_Exception

5. Theory of Operation

The PayPal service object will be used in slightly different manners, depending on the API call that is preformed. Generally speaking, a single interface object is instantiated, and the API credentials are set (API user name, password and signature / certificate). Internally, the object also instantiates a Zend_Http_Client object and prepares it for an NVP request (set the method to POST, preset some always-required POST parameters, etc.).

Depending on the request type, the user will also need to instantiate additional container objects - for example, a DoDirectPayment request would require credit card information and billing address to be passed. These pieces of information will be represented in objects that will be passed to the interface object when calling an API method.

Once an API method is called, the interface object will set all the required POST parameters, and send the HTTP request. All API calls will return a common Response object - containing all the information in the PayPal response (status, transaction ID, failure reasons, etc.)

Note: Supporting PayPal certificate authentication might require improvements to Zend_Http_Client - namely resolving ZF-1004.

6. Milestones / Tasks

- Milestone 1: design notes will be published here
- Milestone 2: As part of the proposal process, integration with other Framework components will be considered (eg. Zend_Currency, Zend_Auth, Zend_Log)
- Milestone 3: Working prototype for DoDirectPayment and ExpressCheckout calls, along with container classes for common data objects

- Milestone 4: Unit tests exist, work, and are checked into SVN.
- Milestone 5: Initial documentation exists.
- Milestone 6: All API other listed API calls are working using the NVP interface, along with unit tests and documentation.
- Milestone 7: (optional, depending on a separate proposal) Implementation of the SOAP interface
- Milestone 8: (optional, depending on a separate proposal) Implementation of a PayPal Instant Payment Notification gateway

7. Class Index

- General
  - Zend_Service_PayPal_Exception
- PayPal Business API access
  - Zend_Service_PayPal_Interface
  - Zend_Service_PayPal_Nvp
  - Zend_Service_PayPal_Soap (optional, depending on a separate proposal)
  - Zend_Service_PayPal_Response
- Data Containers
  - Zend_Service_PayPal_Data_Exception
  - Zend_Service_PayPal_Data_Address
  - Zend_Service_PayPal_Data_AuthInfo
  - Zend_Service_PayPal_Data_CreditCard
  - Zend_Service_PayPal_Data_User
  - Zend_Service_PayPal_Data_OrderItem
  - Zend_Service_PayPal_Data_MassPayReceiver
- Payment Data Transfers
  - Zend_Service_PayPal_Pdt
  - Zend_Service_PayPal_Pdt_Response
  - Zend_Service_PayPal_Pdt_Exception

8. Use Cases

UC-1: Charging a credit card using the DirectPayment API call

```
$authInfo = new Zend_Service_PayPal_Data_AuthInfo(
'mypaypal_user.name.com',
'PAYPALPASSWORD!@#$',
'abcdefgabcdefgabcdefgabcdefgabcdefgabcdefgabcdefgabcdefgabcdefg.paypal_signature');

$creditCard = new Zend_Service_PayPal_Data_CreditCard( // Credit card info
'Visa',             // Card type
'4111111111111111', // Card number
'Bob',              // Buyer's first name
'Sideshow',         // Buyer's last name
12,                 // Expiry month
2009,               // Expiry year
909);               // CVV2

$billingAddress = new Zend_Service_PayPal_Data_Address( // Billing address
'Mazeh 12 Pinat Lamah',  // Streed address
'Tel Aviv',           // City
'IL',                 // County Code
null,                 // State
12345);               // Zip code

$paypal = new Zend_Service_PayPal_Nvp($authInfo);

$result = $paypal->doDirectPayment('Sale', $creditCard, $billingAddress, 12.53);

if ($result->isSuccess()) {
// Yey!
} else {
// Ohhhh..
}
```

UC-2: Charding a PayPal account through Express Checkout

Start the checkout process:

```
$paypal = new Zend_Service_PayPal_Nvp($authInfo);
$params = array('NOSHIPPING' => 1);

$response = $paypal->setExpressCheckout($amount, $self . '?status=ok', $self .
'?status=cancel', $params);
$_SESSION['last_ammount'] = $amount; // PayPal requires us to send the same amount
when finalizing the transaction

if ($response->isSuccess() && ($token = $response->getValue('TOKEN')) {
// Redirect to PayPal's service
header("Location:
https://www.sandbox.paypal.com/cgi-bin/webscr?cmd=_express-checkout&token=$token");
} else {
// Error
echo 'Error initiating PayPal transaction...';
}
```

After the user is redirected back from PayPal after authorizing the payment:

```
if (isset($_GET['token'])) {
$paypal = new Zend_Service_PayPal_Nvp($authInfo);
$coDetails = $paypal->getExpressCheckoutDetails($_GET['token']);

if ($coDetails->isSuccess() && ($payerId = $coDetails->getValue('PAYERID'))) {
// We could present the user with final information here... but we won't ;)
$response = $paypal->doExpressCheckoutPayment($_GET['token'],
$coDetails->getValue('PAYERID'), $_SESSION['last_ammount']);

if ($response->isSuccess()) {
// We got money!
}
}
}
```

UC-3: Preforming a Mass Payment to customers

```
$paypal = new Zend_Service_PayPal_Nvp($authInfo);

// Get list of users I want to pay to
$winners = LotteryWinners::getDailyWinners();

// Build array of Mass Payment Receivers
$rcpts = array();
foreach ($winners as $winner) {
$rcpt = new Zend_Service_PayPal_Data_MassPayReceiver(
$winner->getPrizeAmmount(), // float prize ammount
$winner->getEmailAddress(), // Email address of recepient
Zend_Service_PayPal_Data_MassPayReceiver::RT_EMAIL // Receiver ID type constant
);

$rcpt->setUniqueId($winner->getTransactionId()); // Unique user transaction ID
(optional)
$rcpt->setNote("For winning in the bla bla bla ....") // User-specific note to be
added to e-mail from PayPal

$rcpts[] = $rcpt;
}

// Send MassPay Request
$response = $paypal->doMassPay(
$rcpts,                                                          // Recipients
Zend_Service_PayPal_Data_MassPayReceiver::RT_EMAIL,              // Receiver ID
type
'Congradulations! You just got your prize money from notafraud.com', // Email subject
'USD'                                                            // Currency Code
);

// ... Proceed to verify the response
```

UC-4: Getting information about a transaction using GetTransactionDetails

```
$paypal = new Zend_Service_PayPal_Nvp($authInfo);

// Query about a specific transaction
$response = $paypal->doGetTransactionDetails('1234567890ABCDEFG');

// Display transaction's payer ID
if ($response->isSuccess()) {
echo "Payer ID: " . $response->getValue('PAYERID') . "\n";
}
```

9. Class Skeletons

```
class Zend_Service_PayPal_Exception extends Zend_Service_Exception {}
```

```
class Zend_Service_PayPal_Data_Exception extends Zend_Service_PayPal_Exception {}
```

```
class Zend_Service_PayPal_Nvp
{
/**
* Paypal NVP service URL
*
*/
const SERVICE_URI = 'https://api-3t.sandbox.paypal.com/nvp';

/**
* Authentication info for the PayPal API service
*
* @var Zend_Service_PayPal_Data_AuthInfo
*/
protected $authInfo = null;

/**
* Zend_Http_Client to use for the service
*
* @var Zend_Http_Client
*/
protected $httpClient = null;

/**
* Enter description here...
*
* @param Zend_Service_PayPal_Data_AuthInfo $auth_info
* @param Zend_Http_Client                  $httpClient
*/
public function __construct(Zend_Service_PayPal_Data_AuthInfo $authInfo, $httpClient =
null);
```

```
/**
* HTTP client preparation procedures - should be called before every API
* call.
*
* Will clean up the HTTP client parameters, set the request method to POST
* and add the always-required authentication information
*
* @param string $method The API method we are about to use
* @return void
*/
protected function prepare($method);

/**
* Preform the request and return a response object
*
* @return Zend_Service_PayPal_Response
*/
protected function process();

/**
* Get the HTTP client to be used for this service
*
* @return Zend_Http_Client
*/
public function getHttpClient();

/**
* Preform a DoDirectPayment call
*
* This call preforms a direct payment, directly charging a credit card
* using PayPal's services. It does not require a valid PayPal user name,
* but does require the credit card details and billing address of the
* customer.
*
* @param  Zend_Service_PayPal_Data_CreditCard $creditCard
* @param  Zend_Service_PayPal_Data_Address    $address
* @param  float                               $ammount
* @param  string                              $remoteAddr
* @param  string                              $paymentAction
* @param  array                               $params


* @return Zend_Service_PayPal_Response
*/
public function doDirectPayment($creditCard, $address, $ammount, $remoteAddr = null,
$paymentAction = 'Sale', $params = array());

/**
* Preform a SetExpressCheckout PayPal API call, starting an Express
* Checkout process.
*
* This call is expected to return a token which can be used to redirect
* the user to PayPal's transaction approval page
*
* @param  float  $ammount
* @param  string $returnUrl
* @param  string $cancelUrl
* @param  array  $params    Additional parameters
```

```
* @return Zend_Service_PayPal_Response
*/
public function setExpressCheckout($ammount, $returnUrl, $cancelUrl, $params =
array());

/**
* Preform a GetExpressCheckoutDetails PayPal API call, requesting info
* about a started Express Checkout transaction
*
* @param  string $token Transaction identifier token
* @return Zend_Service_PayPal_Response
*/
public function getExpressCheckoutDetails($token);

/**
* Preform a DoExpressCheckoutPayment PayPal API call, finalizing a
* transaction.
*
* @param  string $token
* @param  string $payerId
* @param  string $ammount
* @param  string $paymentAction Payment action - 'Sale' or 'Authorization'
* @return Zend_Service_PayPal_Response
*/
public function doExpressCheckoutPayment($token, $payerId, $ammount, $paymentAction =
'Sale');

/**
* Preform a Mass Payment API call
*
* @param  array  $receivers    Array of Zend_Service_PayPal_Data_MassPayReceiver
objects
* @param  string $rcpttype     Receiver type
* @param  string $emailSubject Email subject for all receivers
* @param  string $currency     3 letter currency code, default is USD
* @return Zend_Service_PayPal_Response
*/
public function doMassPay(array $receivers, $rcpttype =
Zend_Service_PayPal_Data_MassPayReceiver::RT_EMAIL,
$emailSubject = '', $currency = 'USD');
/**
* Preform a Get Transaction Details API call
*
* @param  string $transactionId Transaction ID (17 Alphanumeric single-byte
characters)
* @return Zend_Service_PayPal_Response
*/
```

```
public function doGetTransactionDetails($transactionId);
}
```

```
class Zend_Service_PayPal_Data_Address
{
/**
* Street
*
* @var string
*/
protected $street;

/**
* City
*
* @var string
*/
protected $city;

/**
* State (2 character code)
*
* @var string
*/
protected $state = 'ZZ';

/**
* Country (2 character code)
*
* @var string
*/
protected $countryCode;

/**
* Zip code
*
* @var integer
*/
protected $zip;

public function __construct($street, $city, $countryCode, $state = null, $zip = null);

/**
* Get the value of street
*
* @return string
*/
public function getStreet();

/**
* Get the value of city
*
* @return string
*/
public function getCity();
```

```
/**
* Get the value of state
*
* @return string
*/
public function getState();

/**
* Get the value of countryCode
*
* @return string
*/
public function getCountryCode();

/**
* Get the value of zip
*
* @return integer
*/
```

```
public function getZip();
}
```

```
class Zend_Service_PayPal_Response
{
protected $_data = array();

/**
* Create a new response object from response string
*
* @param string $data
*/
public function __construct($response);

/**
* Check whether the request was successfull
*
* @return boolean
*/
public function isSuccess();

/**
* Check whether the request failed
*
* @return boolean
*/
public function isFailure();

/**
* Check whether the request produced warnings
*
* Note: Successful requests might produce warnings as well
*
* @return boolean
*/
public function hasWarnings();

/**
* Get the value of a response parameter
*
* @param  string $name
* @return string|null Value or null if not set
*/
public function getValue($name);

/**
* Magic wrapper around getValue()
*
* Unlike getValue(), accessing parameters through __get() will throw an exception if
the
* parameter was not set in the response.
*
* @param  string $name Key
* @return string
*/
public function __get($name);
}
```

```
class Zend_Service_PayPal_Data_MassPayReceiver
{
/**
* Receiver type constants
*/
const RT_EMAIL  = 'EmailAddress';  // Receiver identified by e-mail address
const RT_USERID = 'UserID';        // Receiver identified by PayPal user ID

/**
* Receiver identifier (email address or PayPal ID)
*
* @var string
*/
protected $receiverid   = null;

/**
* Receiver ID type - one of the two type constants
*
* @var string
*/
protected $receivertype = null;

/**
* Ammount to pay (currency is defined in transaction)
*
* @var float
*/
protected $ammount      = null;

/**
* Optional unique transaction ID
*
* @var string
*/
protected $uniqueid     = null;

/**
* Optional customer-specific note
*
* @var string
*/
protected $note         = null;

/**
* Create a new receiver info object
*
* @param float  $amount   Amount to pay (> 0)
* @param string $rcpt      Unique receiver identifier
* @param string $rcpt_type One of the two type constants
*/
public function __construct($amount, $rcpt, $rcpt_type = self::RT_EMAIL);

/**
* Set the optional unique receiver ID
*
* @param  string $id
* @return Zend_Service_PayPal_Data_MassPayReceiver
*/
```

```
public function setUniqueId($id);

/**
* Get the optional recipient unique ID
*
* @return string
*/
public function getUniqueId();

/**
* Set receiver specific custom note
*
* @param  string $note
* @return Zend_Service_PayPal_Data_MassPayReceiver
*/
public function setNote($note);

/**
* Return the customer-specific optional custom note
*
* @return string
*/
public function getNote();

/**
* Get the amount to pay (currency is determined by the transaction)
*
* @return float
*/
public function getAmount();

/**
* Get the receiver ID (email address or PayPal ID)
*
* @return string
*/
public function getReceiverId();

/**
* Get the receiver type (must match transaction-global type)
*
* @return string
*/
public function getReceiverType();

/**
* Validate that the receiver ID is well-formed according to it's type
*
* @param  string $value
* @param  string $type  Either ::RT_EMAIL or ::RT_USERID
* @return boolean
*/
```

```
static public function validateReceiverType($value, $type);
}
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>